Online Object Tracking, Learning and Parsing with And-Or Graphs

Tianfu Wu, Yang Lu and Song-Chun Zhu

Abstract—This paper presents a method, called *AOGTracker*, for simultaneously tracking, learning and parsing (TLP) unknown objects in video sequences with a hierarchical and compositional And-Or graph (AOG) representation. The TLP method is formulated in the Bayesian framework with a spatial and a temporal dynamic programming (DP) algorithms inferring object bounding boxes on-the-fly. During online learning, the AOG is discriminatively learned using latent SVM [1] to account for appearance (e.g., lighting and partial occlusion) and structural (e.g., different poses and viewpoints) variations of a tracked object, as well as distractors (e.g., similar objects) in background. Three key issues in online inference and learning are addressed: (i) maintaining purity of positive and negative examples collected online, (ii) controling model complexity in latent structure learning, and (iii) identifying critical moments to re-learn the structure of AOG based on its intrackability. The intrackability measures uncertainty of an AOG based on its score maps in a frame. In experiments, our AOGTracker is tested on two popular tracking benchmarks with the same parameter setting: the TB-100/50/CVPR2013 benchmarks [2], [3], and the VOT benchmarks [4] — VOT 2013, 2014, 2015 and TIR2015 (thermal imagery tracking). In the former, our AOGTracker outperforms state-of-the-art tracking algorithms including two trackers based on deep convolutional network [5], [6]. In the latter, our AOGTracker outperforms all other trackers in VOT2013 and is comparable to the state-of-the-art methods in VOT2014, 2015 and TIR2015.

Index Terms—Visual Tracking, And-Or Graphs, Latent SVM, Dynamic Programming, Intrackability

1 Introduction

1.1 Motivation and Objective

Online object tracking is an innate capability in human and animal vision for learning visual concepts [7], and is an important task in computer vision. Given the state of an unknown object (e.g., its bounding box) in the first frame of a video, the task is to infer hidden states of the object in subsequent frames. Online object tracking, especially long-term tracking, is a difficult problem. It needs to handle variations of a tracked object, including appearance and structural variations, scale changes, occlusions (partial or complete), etc. It also needs to tackle complexity of the scene, including camera motion, background clutter, distractors, illumination changes, frame cropping, etc. Fig. 1 illustrates some typical issues in online object tracking. In recent literature, object tracking has received much attention due to practical applications in video surveillance, activity and event prediction, human-computer interactions and traffic monitoring.

This paper presents an integrated framework for online tracking, learning and parsing (TLP) unknown objects with a unified representation. We focus on settings in which object state is represented by bounding box, without using pre-trained models. We address five issues associated with online object tracking in the following.

Issue I: Expressive representation accounting for structural and appearance variations of unknown objects in tracking. We are interested in hierarchical and compositional object models. Such

Fig. 1: Illustration of some typical issues in online object tracking using the "skating1" video in the benchmark [2]. Starting from the object specified in the first frame, a tracker needs to handle many variations in subsequent frames which include *illuminative variation, scale variation, occlusion, deformation, fast motion, in-plane and out-of-plane rotation, background clutter*, etc.

models have shown promising performance in object detection [1], [8], [9], [10], [11] and object recognition [12]. A popular modeling scheme represents object categories by mixtures of deformable part-based models (DPMs) [1]. The number of mixture components is usually predefined and the part configuration of each component is fixed after initialization or directly based on strong supervision. In online tracking, since a tracker can only access the ground-truth object state in the first frame, it is not suitable for it to "make decisions" on the number of mixture components and part configurations, and it does not have enough data to learn. It's desirable to have an object representation which has expressive power to represent a large number of part configurations, and can facilitate computationally effective inference and learning. We quantize the space of part configurations recursively in a principled way with a hierarchical and compositional And-Or graph (AOG) representation [8], [11]. We learn and update the most discriminative part configurations online by pruning the quantized space based on discriminative power.

Issue II: Computing joint optimal solutions. Online object

- *T.F. Wu and Y. Lu are with the Department of Statistics, University of California, Los Angeles.*
  E-mail: tfwu@stat.ucla.edu, yanglv@ucla.edu
- *S.-C. Zhu is with the Department of Statistics and Computer Science, University of California, Los Angeles.*
  E-mail: sczhu@stat.ucla.edu

Manuscript received MM DD, YYYY; revised MM DD, YYYY.

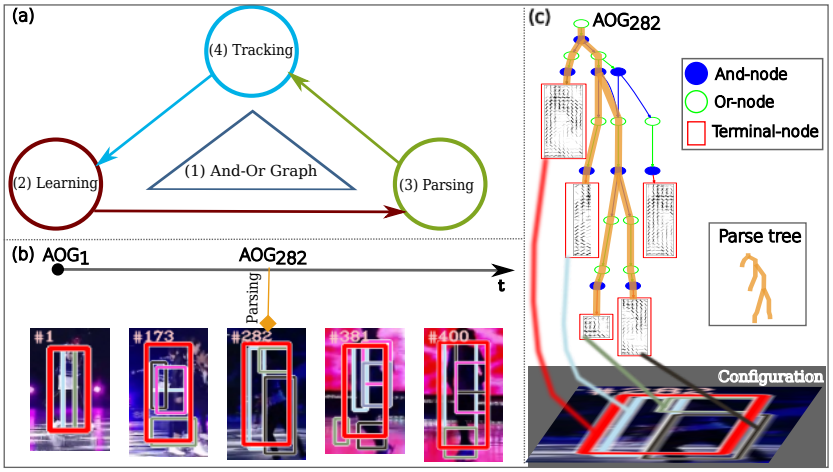


Fig. 2: Overview of our AOGTracker. (a) Illustration of the tracking, learning and parsing (TLP) framework. It consists of four components. (b) Examples of capturing structural and appearance variations of a tracked object by a series of object configurations inferred on-the-fly over key frames #1, #173, #282, etc. (c) Illustration of an object AOG, a parse tree and an object configuration in frame #282. A parse tree is an instantiation of an AOG. A configuration is a layout of latent parts represented by terminal-nodes in a parse tree. An object AOG preserves ambiguities by capturing multiple parse trees.

tracking is usually posed as a maximum a posterior (MAP) problem using first order hidden Markov models (HMMs) [2], [13], [14]. The likelihood or observation density is temporally inhomogeneous due to online updating of object models. Typically, the objective is to infer the most likely hidden state of a tracked object in a frame by maximizing a Bayesian marginal posterior probability given all the data observed so far. The maximization is based on either particle filtering [15] or dense sampling such as the tracking-by-detection methods [16], [17], [18]. In most prior approaches (e.g., the 29 trackers evaluated in the TB-100 benchmark [2]), no feedback inspection is applied to the history of inferred trajectory. We utilize tracking-by-parsing with hierarchical models in inference. We allow feedback inspection by maximizing a Bayesian joint posterior probability of trajectory given all the observation. Thus, a tracker can trace back the trajectory to potentially improve accuracy at each step as more evidence has been observed. By doing so, we simultaneously address another key issue in online learning (Issue III).

Issue III: Maintaining the purity of a training dataset. The dataset consists of a set of positive examples computed based on the current trajectory, and a set of negative examples mined from outside the current trajectory. In the dataset, we can only guarantee that the positives and the negatives in the first frame are true positives and true negatives respectively. A tracker needs to carefully choose frames from which it can learn to avoid model drifting (i.e., self-paced learning). Most prior approaches do not address this issue since they focus on marginally optimal solutions with which object models are updated, except for the P-N learning in TLD [17] and the self-paced learning for tracking [18]. Since we allow feedback inspection in tracking, we can correct previous errors in the training dataset.

Issue IV: Failure-aware online learning of object models. In online learning, we mostly update model parameters incrementally after inference in a frame. Theoretically speaking, after an initial object model is learned in the first frame, model drifting is inevitable in general setting. Thus, in addition to maintaining the purity of a training dataset, it is also important that we can identify critical moments (caused by different structural and appearance variations) automatically. At those moments, a tracker needs to re-learn both the structure and the parameters of object model using the current whole training dataset. We address this issue by computing uncertainty of an object model in a frame based on its response maps.

Issue V: Computational efficiency by dynamic search strategy. Most tracking-by-detection methods run detection in the whole frame since they usually use relatively simple models such as a single object template. With hierarchical models in tracking and sophisticated online inference and updating strategies, the computational complexity is high. To speed up tracking, we need to utilize a dynamic search strategy. This strategy must take into account the trade-off between generating a conservative proposal state space for efficiency and allowing an exhaustive search for accuracy (e.g., to handle the situation where the object is completely occluded for a while or moves out the camera view and then reappears). We address this issue by adopting a simple search cascade with which we run detection in the whole frame only when local search has failed.

Our TLP method obtains state-of-the-art performance on one popular tracking benchmark [2]. We give a brief overview of our method in the next subsection.

1.2 Method Overview

As illustrated in Fig.2 (a), the TLP method consists of four components. We introduce them briefly as follows.

(1) *An AOG quantizing the space of part configurations.* Given the bounding box of an object in the first frame, we assume object parts are also of rectangular shapes. We first divide it evenly into a small cell-based grid (e.g., 3×3) and a cell defines the smallest part. We then enumerate all possible parts with different aspect ratios and different sizes which can be placed inside the grid. All the enumerated parts are organized into a hierarchical and compositional AOG. Each part is represented by a terminal-node. Two types of nonterminal nodes as compositional rules: an And-node represents the decomposition of a large part into two smaller ones, and an Or-node represents alternative ways of decompositions through different horizontal or vertical binary splits. We call it **the full structure AOG**[1]. It is capable of exploring a large number of latent part configurations (see some examples in Fig. 2 (b)), meanwhile it makes the problem of online model learning feasible.

(2) *Learning object AOGs.* An object AOG is a subgraph learned from the full structure AOG (see Fig. 2 (c) [2]). Learning an object AOG consists of two steps: (i) The initial object AOG are learned by pruning branches of Or-nodes in the full structure AOG based on discriminative power, following breadth-first search

1. By full structure, it means all the possible compositions on top of the grid with binary composition being used for And-nodes

2. We note that there are some Or-nodes in the object AOGs which have only one child node since they are subgraphs of the full structure AOG and we keep their original structures.

(BFS) order. The discriminative power of a node is measured based on its training error rate. We keep multiple branches for each encountered Or-node to preserve ambiguities, whose training error rates are not bigger than the minimum one by a small positive value. (ii) We retrain the initial object AOG using latent SVM (LSVM) as it was done in learning the DPMs [1]. LSVM utilizes positive re-labeling (i.e., inferring the best configuration for each positive example) and hard negative mining. To further control the model complexity, we prune the initial object AOG through majority voting of latent assignments in positive re-labeling.

(3) *A spatial dynamic programming (DP) algorithm for computing all the proposals in a frame with the current object AOG*. Thanks to the DAG structure of the object AOG, a DP parsing algorithm is utilized to compute the matching scores and the optimal parse trees of all sliding windows inside the search region in a frame. A parse tree is an instantiation of the object AOG which selects the best child for each encountered Or-node according to matching score. A configuration is obtained by collapsing a parse tree onto the image domain, capturing layout of latent parts of a tracked object in a frame.

(4) *A temporal DP algorithm for inferring the most likely trajectory*. We maintain a DP table memorizing the candidate object states computed by the spatial DP in the past frames. Then, based on the first-order HMM assumption, a temporal DP algorithm is used to find the optimal solution for the past frames jointly with pair-wise motion constraints (i.e., the Viterbi path [14]). The joint solution can help correct potential tracking errors (i.e., false negatives and false positives collected online) by leveraging more spatial and temporal information. This is similar in spirit to methods of keeping N-best maximal decoder for part models [19] and maintaining diverse M-best solutions in MRF [20].

2 Related Work

In the literature of object tracking, either single object tracking or multiple-object tracking, there are often two settings.

Offline visual tracking [21], [22], [23], [24]. These methods assume the whole video sequence has been recorded, and consist of two steps. i) It first computes object proposals in all frames using some pre-trained detectors (e.g., the DPMs [1]) and then form "tracklets" in consecutive frames. ii) It seeks the optimal object trajectory (or trajectories for multiple objects) by solving an optimization problem (e.g., the K-shortest path or min-cost flow formulation) for the data association. Most work assumed first-order HMMs in the formulation. Recently, Hong and Han [25] proposed an offline single object tracking method by sampling tree-structured graphical models which exploit the underlying intrinsic structure of input video in an orderless tracking [26].

Online visual tracking for streaming videos. It starts tracking after the state of an object is specified in certain frame. In the literature, particle filtering [15] has been widely adopted, which approximately represents the posterior probability in a non-parametric form by maintaining a set of particles (i.e., weighted candidates). In practice, particle filtering does not perform well in high-dimensional state spaces. More recently, tracking-by-detection methods [16], [17] have become popular which learn and update object models online and encode the posterior probability using dense sampling through sliding-window based detection on-the-fly. Thus, object tracking is treated as instance-based object detection. To leverage the recent advance in object detection, object tracking research has made progress by incorporating discriminatively trained part-based models [1], [8], [27] (or more generally grammar models [9], [10], [11]). Most popular methods also assume first-order HMMs except for the recently proposed online graph-based tracker [28]. There are four streams in the literature of online visual tracking:

i) Appearance modeling of the whole object, such as incremental learning [29], kernel-based [30], particle filtering [15], sparse coding [31] and 3D-DCT representation [32]; More recently, Convolutional neural networks are utilized in improving tracking performance [5], [6], [33], which are usually pre-trained on some large scale image datasets such as the ImageNet [34] or on video sequences in a benchmark with the testing one excluded.
ii) Appearance modeling of objects with parts, such as patch-based [35], coupled 2-layer models [36] and adaptive sparse appearance [37]. The major limitation of appearance modeling of a tracked object is the lack of background models, especially in preventing model drift from distracotrs (e.g., players in sport games). To address this issue, it leads to discriminant tracking;
iii) Tracking by discrimination using a single classifier, such as support vector tracking [38], multiple instance learning [39], STRUCK [40], circulant structure-based kernel method [41], and discriminant saliency based tracking [42];
iv) Tracking by part-based discriminative models, such as online extensions of DPMs [43], and structure preserving tracking method [27], [44].

Our method belongs to the fourth stream of online visual tracking. Unlike predefined or fixed part configurations with star-model structure used in previous work, our method learns both structure and appearance of object AOGs online, which is, to our knowledge, the first method to address the problem of online explicit structure learning in tracking. The advantage of introducing AOG representation are three-fold.

i) *More representational power*: Unlike TLD [17] and many other methods (e.g., [18]) which model an object as a single template or a mixture of templates and thus do not perform well in tracking objects with large structural and appearance variations, an AOG represents an object in a hierarchical and compositional graph expressing a large number of latent part configurations.
ii) *More robust tracking and online learning strategies*: While the whole object has large variations or might be partially occluded from time to time during tracking, some other parts remain stable and are less likely to be occluded. Some of the parts can be learned to robustly track the object, which can also improve accuracy of appearance adaptation of terminal-nodes. This idea is similar in spirit to finding good features to track objects [45], and we find good part configurations online for both tracking and learning.
iii) *Fine-grained tracking results*: In addition to predicting bounding boxes of a tracked object, outputs of our AOGTracker (i.e., the parse trees) have more information which are potentially useful for other modules beyond tracking such as activity or event prediction.

Our preliminary work has been published in [46] and the method for constructing full structure AOG was published in [8]. This paper extends them by: (i) adding more experimental results with state-of-the-art performance obtained and full source code released; (ii) elaborating details substantially in deriving the formulation of inference and learning algorithms; and (iii) adding

more analyses on different aspects of our method. This paper makes three contributions to the online object tracking problem:

i) It presents a tracking-learning-parsing (TLP) framework which can learn and track objects AOGs.
ii) It presents a spatial and a temporal DP algorithms for tracking-by-parsing with AOGs and outputs fine-grained tracking results using parse trees.
iii) It outperforms the state-of-the-art tracking methods in a recent public benchmark, TB-100 [2], and obtains comparable performance on a series of VOT benchmarks [4].

Paper Organization. The remainder of this paper is organized as follows. Section 3 presents the formulation of our TLP framework under the Bayesian framework. Section 4 gives the details of spatial-temporal DP algorithm. Section 5 presents the online learning algorithm using the latent SVM method. Section 6 shows the experimental results and analyses. Section 7 concludes this paper and discusses issues and future work.

3 Problem Formulation

3.1 Formulation of Online Object Tracking

In this section, we first derive a generic formulation from generative perspective in the Bayesian framework, and then derive the discriminative counterpart.

3.1.1 Tracking with HMM

Let Λ denote the image lattice on which video frames are defined. Denote a sequence of video frames within time range $[1, T]$ by,

$$I_{1:T} = \{I_1, \cdots, I_T\}. \tag{1}$$

Denote by B_t the bounding box of a target object in I_t. In online object tracking, B_1 is given and B_t's are inferred by a tracker ($t \in [2, T]$). With first-order HMM, we have,

$$\text{The prior model:} \quad B_1 \sim p(B_1)\,, \tag{2}$$
$$\text{The motion model:} \quad B_t|B_{t-1} \sim p(B_t|B_{t-1})\,, \tag{3}$$
$$\text{The likelihood:} \quad I_t|B_t \sim p(I_t|B_t). \tag{4}$$

Then, the prediction model is defined by,

$$p(B_t|I_{1:t-1}) = \int_{\Omega_{B_{t-1}}} p(B_t|B_{t-1})p(B_{t-1}|I_{1:t-1})dB_{t-1}, \tag{5}$$

where $\Omega_{B_{t-1}}$ is the candidate space of B_{t-1}, and the updating model is defined by,

$$p(B_t|I_{1:t}) = p(I_t|B_t)p(B_t|I_{1:t-1})/p(I_t|I_{1:t-1}), \tag{6}$$

which is a marginal posterior probability. The tracking result, the best bounding box B_t^*, is computed by,

$$B_t^* = \arg\max_{B_t \in \Omega_{B_t}} p(B_t|I_{1:t}), \tag{7}$$

which is usually solved using particle filtering [15] in practice.

To allow feedback inspection of the history of a trajectory, we seek to maximize a joint posterior probability,

$$\begin{aligned} p(B_{1:t}|I_{1:t}) &= p(B_{1:t-1}|I_{1:t-1})\frac{p(B_t|B_{t-1})p(I_t|B_t)}{p(I_t|I_{1:t-1})} \\ &= p(B_1|I_1)\prod_{i=2}^{t}\frac{p(B_i|B_{i-1})p(I_i|B_i)}{p(I_i|I_{1:i-1})}. \end{aligned} \tag{8}$$

By taking the logarithm of both sides of Eqn.(8), we have,

$$\begin{aligned} B_{1:t}^* &= \arg\max_{B_{1:t}} \log p(B_{1:t}|I_{1:t}) \\ &= \arg\max_{B_{1:t}} \{\log p(B_1) + \log p(I_1|B_1) + \\ &\qquad \sum_{i=2}^{t}[\log p(B_i|B_{i-1}) + \log p(I_i|B_i)]\}. \end{aligned} \tag{9}$$

where the image data term $p(I_1)$ and $\sum_{i=2}^{t} p(I_i|I_{1:i-1})$ are not included in the maximization as they are treated as constant terms.

Since we have ground-truth for B_1, $p(I_1|B_1)$ can also be treated as known after the object model is learned based on B_1. Then, Eqn.(9) can be reproduced as,

$$\begin{aligned} B_{2:t}^* &= \arg\max_{B_{2:t}} \log p(B_{2:t}|I_{1:t}, B_1) \\ &= \arg\max_{B_{2:t}} \{\sum_{i=2}^{t}[\log p(B_i|B_{i-1}) + \log p(I_i|B_i)]\}. \end{aligned} \tag{10}$$

3.1.2 Tracking as Energy Minimization over Trajectories

To derive the discriminative formulation of Eqn.(10), we show that only the log-likelihood ratio matters in computing $\log p(I_i|B_i)$ in Eqn.(10) with very mild assumptions.

Let Λ_{B_i} be the image domain occupied by a tracked object, and $\overline{\Lambda_{B_i}}$ the remaining domain (i.e., $\overline{\Lambda_{B_i}} \cup \Lambda_{B_i} = \Lambda$ and $\overline{\Lambda_{B_i}} \cap \Lambda_{B_i} = \emptyset$) in a frame I_i. With the independence assumption between $I_{\Lambda_{B_i}}$ and $I_{\overline{\Lambda_{B_i}}}$ given B_i, we have,

$$\begin{aligned} p(I_i|B_i) &= p(I_{\Lambda_{B_i}}, I_{\overline{\Lambda_{B_i}}}|B_i) = p(I_{\Lambda_{B_i}}|B_i)p(I_{\overline{\Lambda_{B_i}}}|B_i) \\ &= p(I_{\Lambda_{B_i}}|B_i)q(I_{\overline{\Lambda_{B_i}}}) = q(I_\Lambda)\frac{p(I_{\Lambda_{B_i}}|B_i)}{q(I_{\Lambda_{B_i}})}, \end{aligned} \tag{11}$$

where $q(I_\Lambda)$ is the probability model of background scene and we have $q(I_{\overline{\Lambda_{B_i}}}) = p(I_{\overline{\Lambda_{B_i}}}|B_i)$ w.r.t. context-free assumption. So, $q(I_\Lambda)$ does not need to be specified explicitly and can be omitted in the maximization. This derivation gives an alternative explanation for discriminant tracking *v.s.* tracking by generative appearance modeling of an object [47].

Based on Eqn.(10), we define an energy function by,

$$\mathcal{E}(B_{2:t}|I_{1:t}, B_0) \propto -\log p(B_{2:t}|I_{1:t}, B_1). \tag{12}$$

And, we do not compute $\log p(I_i|B_i)$ in the probabilistic way, instead we compute matching score defined by,

$$\begin{aligned} \text{Score}(I_i|B_i) &= \log\frac{p(I_{\Lambda_{B_i}}|B_i)}{q(I_{\Lambda_{B_i}})} \\ &= \log p(I_i|B_i) - \log q(I_\Lambda). \end{aligned} \tag{13}$$

which we can apply discriminative learning methods.

Also, denote the motion cost by,

$$\text{Cost}(B_i|B_{i-1}) = -\log p(B_i|B_{i-1}). \tag{14}$$

We use a thresholded motion model in experiments: the cost is 0 if the transition is accepted based on the median flow [17] (which is a forward-backward extension of the Lucas-Kanade optimal flow [48]) and $+\infty$ otherwise. A similar method was experimented in [18].

So, we can re-write Eqn.(10) in the minimization form,

$$\begin{aligned} B_{2:t}^* &= \arg\min_{B_{2:t}} \mathcal{E}(B_{2:t}|I_{1:t}, B_1) \\ &= \arg\min_{B_{2:t}} \{\sum_{i=2}^{t}[\text{Cost}(B_i|B_{i-1}) - \text{Score}(I_i|B_i)]\}. \end{aligned} \tag{15}$$

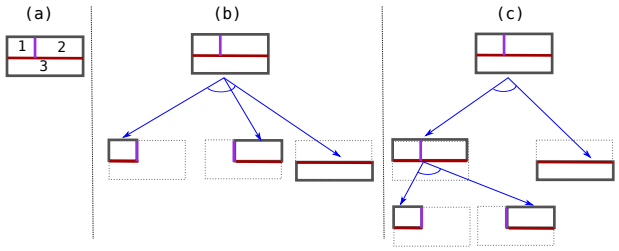


Fig. 3: We assume parts are of rectangular shapes. (a) shows a configuration with 3 parts. Two different, yet equivalent, decomposition rules in representing a configuration are shown in (b) for decomposition with branching factor equal to the number of parts (i.e., a flat structure), and in (c) for a hierarchical decomposition with branching factor being set to 2 at all levels.

In our TLP framework, we compute Score$(I_i|B_i)$ in Eqn.(15) with an object AOG. So, we interpret a sliding window by the optimal parse tree inferred from object AOG. We treat parts as latent variables which are modeled to leverage more information for inferring object bounding box. We note that we do not track parts explicitly in this paper.

3.2 Quantizing the Space of Part Configurations

In this section, we first present the construction of a full structure AOG which quantizes the space of part configurations. We then introduce notations in defining an AOG.

Part configurations. For an input bounding box, a part configuration is defined by a partition with different number of parts of different shapes (see Fig. 3 (a)). Two natural questions arise: (i) How many part configurations (i.e., the space) can be defined in a bounding box? (ii) How to organize them into a compact representation? Without posing some structural constraints, it is a combinatorial problem.

We assume rectangular shapes are used for parts. Then, a configuration can be treated as a tiling of input bounding box using either horizontal or vertical cuts. We utilize binary splitting rule only in decomposition (see Fig. 3 (b) and (c)). With these two constraints, we represent all possible part configurations by a hierarchical and compositional AOG constructed in the following.

Given a bounding box, we first divide it evenly into a cell-based grid (e.g., 9×10 grid in the right of Fig. 4). Then, in the grid, we define a dictionary of part types and enumerate all instances for all part types.

A dictionary of part types. A part type is defined by its width and height. Starting from some minimal size (such as 2×2 cells), we enumerate all possible part types with different aspect ratios and sizes which fit the grid (see A, B, C, D in Fig.4 (a)).

Part instances. An instance of a part type is obtained by placing the part type at a position. Thus, a part instance is defined by a "sliding window" in the grid. Fig.4 (b) shows an example of placing part type D (2×5 cells) in a 9×10 grid with 48 instances in total.

To represent part configurations compactly, we exploit the compositional relationships between enumerated part instances.

The full structure AOG. For any sub-grid indexed by the left-top position, width and height (e.g., $(2, 3, 5, 2)$ in the right-middle of Fig.4 (c)), we can either terminate it directly to the corresponding part instance (Fig.4 (c.1)), or decompose it into two smaller sub-grids using *either horizontal or vertical binary splits*. Depending on the side length, we may have multiple valid splits

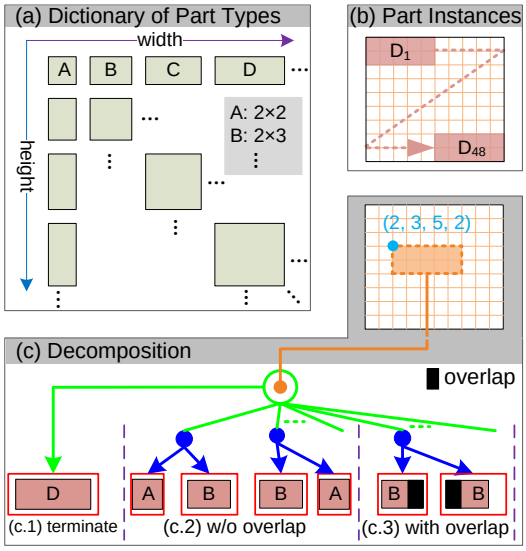


Fig. 4: Illustration of (a) the dictionary of part types, and (b) part instances generated by placing a part type in a grid. Given part instances, (c) shows how a sub-grid is decomposed in different ways. We allow overlap between child nodes (see (c.3)).

along both directions (Fig.4 (c.2)). When splitting either side we allow overlaps between the two sub-grids up to some ratio (Fig.4 (c.3)). Then, we represent the sub-grid as an Or-node, which has a set of child nodes including a terminal-node (i.e. the part instance directly terminated from it), and a number of And-nodes (each of which represents a valid decomposition). This procedure is applied recursively for all child sub-grids. Starting from the whole grid and using BFS order, we construct a full structure AOG (see Fig. 5 for an example). Table. 1 lists the number of part configurations for three cases from which we can see that full structure AOGs cover a large number of part configurations using a relatively small set of part instances.

We denote an AOG by,

$$\mathcal{G} = (V_{And}, V_{Or}, V_T, E, \Theta) \tag{16}$$

where V_{And}, V_{Or} and V_T represent a set of And-nodes, Or-nodes and terminal-nodes respectively, E a set of edges and Θ a set of parameters (to be defined in Section 4.1). We have,

i) *The object/root Or-node (plotted by green circles)*, which represents alternative object configurations;
ii) *A set of And-nodes (solid blue circles)*, each of which represents the rule of decomposing a complex structure (e.g., a walking person or a running basketball player) into simpler ones;
iii) *A set of part Or-nodes*, which handle local variations and configurations in a recursive way;
iv) *A set of terminal-nodes (red rectangles)*, which link an object and its parts to image data (i.e., grounding symbols) to account for appearance variations and occlusions (e.g., head-shoulder of a walking person before and after opening a sun umbrella).

An object AOG is a subgraph of a full structure AOG with the same root Or-node. For notational simplicity, we also denote by $\mathcal{G}$ an object AOG. So, we will write Score$(I_i|B_i; \mathcal{G})$ in Eqn.(15) with $\mathcal{G}$ added.

A parse tree is an instantiation of an object AOG with the best child node (w.r.t. matching scores) selected for each encountered Or-node. All the terminal-nodes in a parse tree represents a part configuration when collapsed to image domain.

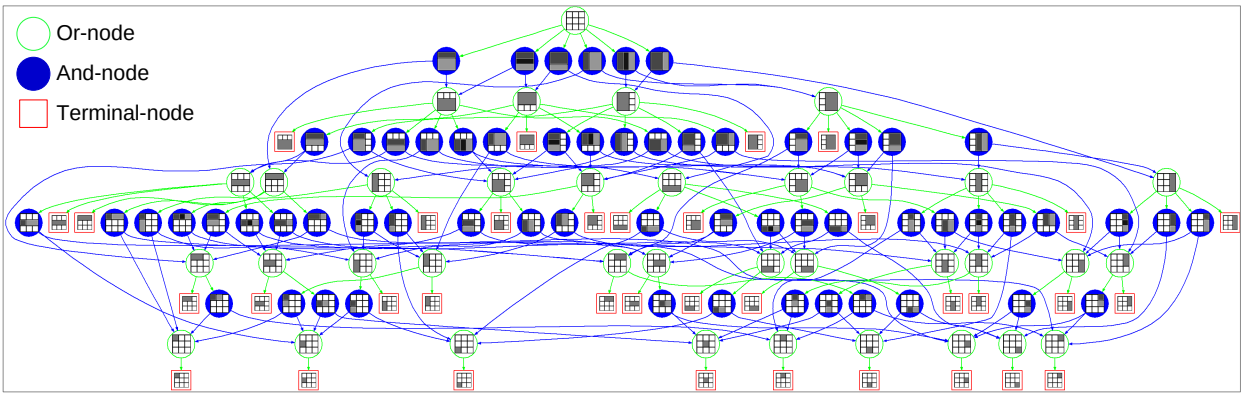


Fig. 5: Illustration of full structure And-Or Graph (AOG) representing the space of part configurations. It is of directed acyclic graph (DAG) structure. For clarity, we show a toy example constructed for a 3×3 grid. The AOG can generate all possible part configurations (the number is often huge for typical grid sizes, see Table.1), while allowing efficient exploration with a DP algorithm due to the DAG structure. See text for details. (Best viewed in color and with magnification)

Grid	primitive part	#Configuration	#T-node	#And-node
3×3	1×1	319	35	48
5×5	1×1	76,879,359	224	600
10×12	2×2	3.8936e+009	1409	5209

TABLE 1: The number of part configurations generated from our AOG without considering overlapped compositions.

We note that an object AOG contains multiple parse trees to preserve ambiguities in interpreting a tracked object (see examples in Fig. 2 (c) and Fig. 7).

4 Tracking-by-Parsing with Object AOGs

In this section, we present details of inference with object AOGs. We first define scoring functions of nodes in an AOG. Then, we present a spatial DP algorithm for computing $\text{Score}(I_i|B_i;\mathcal{G})$, and a temporal DP algorithm for inferring the trajectory $B^*_{2:t}$ in Eqn.(15).

4.1 Scoring Functions of Nodes in an AOG

Let $\mathbb{F}$ be the feature pyramid computed for either the local ROI or the whole image I_t, and $\mathbf{\Lambda}$ the position space of pyramid $\mathbb{F}$. Let $p = (l, x, y) \in \mathbf{\Lambda}$ specify a position (x, y) in the l-th level of pyramid $\mathbb{F}$.

Given an AOG $\mathcal{G} = (V_T, V_{And}, V_{Or}, E, \Theta)$ (e.g., the left in Fig.6), we define four types of edges, i.e., $E = E_T \cup E_{Def} \cup E_{Dec} \cup E_{Switch}$ as shown in Fig.6. We elaborate the definitions of parameters $\Theta = (\Theta^{app}, \Theta^{def}, \Theta^{bias})$:

i) Each terminal-node $\mathbf{t} \in V_T$ has appearance parameters $\theta^{app}_{\mathbf{t}} \subset \Theta^{app}$, which is used to ground a terminal-node to image data.

i) The parent And-node A of a part terminal-node with deformation edge has deformation parameters $\theta^{def}_A \subset \Theta^{def}$. They are used for penalizing local displacements when placing a terminal-node around its anchor position. We note that the object template is not allowed to perturb locally in inference, so the parent And-node of the object terminal-node does not have deformation parameters.

iii) A child And-node of the root Or-node has a bias term $\Theta^{bias} = \{b\}$. We do not define bias terms for child nodes of other Or-nodes.

Appearance Features. We use three types of features: histogram of oriented gradient (HOG) [49], local binary pattern features (LBP) [50], and RGB color histograms (for color videos).

Deformation Features. Denote by $\delta = [dx, dy]$ the displacement of placing a terminal-node around its anchor location. The deformation feature is defined by $\Phi^{def}(\delta) = [dx^2, dx, dy^2, dy]'$ as done in DPMs [1].

We use linear functions to evaluate both appearance scores and deformation scores. The score functions of nodes in an AOG are defined as follows:

i) For a terminal-node $\mathbf{t}$, its score at a position p is computed by,

$$\text{Score}(\mathbf{t}, p|\mathbb{F}) =< \theta^{app}_{\mathbf{t}}, \mathbb{F}(\mathbf{t}, p) > \tag{17}$$

where $< \cdot, \cdot >$ represents inner product and $\mathbb{F}(\mathbf{t}, p)$ extracts features in feature pyramid.

ii) For an Or-node O, its score at position p takes the maximum score over its child nodes,

$$\text{Score}(O, p|\mathbb{F}) = \max_{c \in ch(O)} \text{Score}(c, p|\mathbb{F}) \tag{18}$$

where $ch(v)$ denotes the set of child nodes of a node v.

iii) For an And-node A, we have three different functions w.r.t. the type of its out-edge (i.e., Terminal-, Deformation-, or Decomposition-edge),

$$\text{Score}(A, p|\mathbb{F}) = \tag{19}$$
$$\begin{cases} \text{Score}(\mathbf{t}, p|\mathbb{F}), & e_{A,\mathbf{t}} \in E_T \\ \max_\delta[\text{Score}(\mathbf{t}, p \oplus \delta|\mathbb{F}) - < \theta^{def}_A, \Phi^{def}(\delta) >], & e_{A,\mathbf{t}} \in E_{Def} \\ \sum_{c \in ch(A)} \text{Score}(c, p|\mathbb{F}), & e_{A,c} \in E_{Dec} \end{cases}$$

where the first case is for sharing score maps between the object terminal-node and its parent And-node since we do not allow local deformation for the whole object, the second case for computing transformed score maps of parent And-node of a part terminal-node which is allowed to find the best placement through distance transformation [1], $\oplus$ represents the displacement operator in the position space in $\mathbf{\Lambda}$, and the third case for computing the score maps of an And-node which has two child nodes through composition.

4.2 Tracking-by-Parsing

With scoring functions defined above, we present a spatial DP and a temporal DP algorithms in solving Eqn.(15).

Spatial DP: The DP algorithm consists of two stages: *(i) The bottom-up pass* computes score map pyramids (as illustrated in

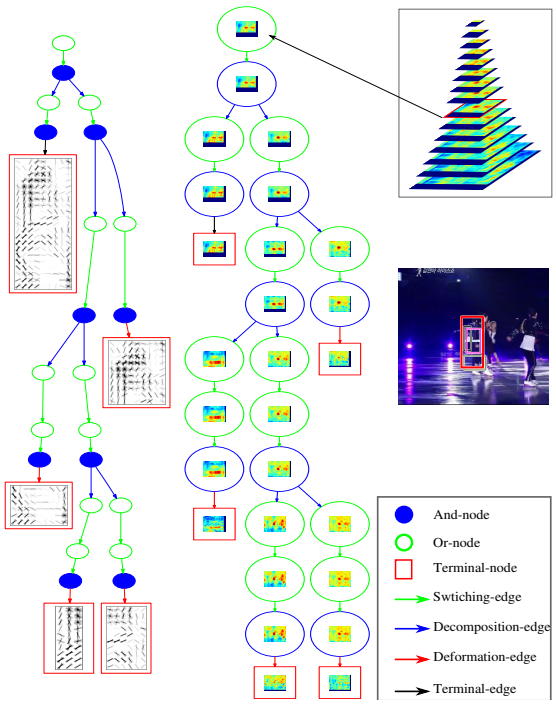


Fig. 6: Illustration of the spatial DP algorithm for parsing with AOGs (e.g., AOG_{172} in the *left*). *Right-middle*: The input image (ROI in the 173-th frame in the "Skating1" sequence) and the inferred object configuration. *Right-top*: The score map pyramid for root Or-node. *Middle*: For each node in AOG, we show one level of score map pyramid at which the optimal parse tree is retrieved.

Fig. 6) for all nodes following the depth-first-search (DFS) order of nodes. It computes matching scores of all possible parse trees at all possible positions in feature pyramid. *(ii) In the top-down pass*, we first find all candidate positions for the root Or-node O based on its score maps and current threshold $\tau_{\mathcal{G}}$ of the object AOG, denoted by

$$\Omega_{cand} = \{p; \text{Score}(O, p|\mathbb{F}) \geq \tau_{\mathcal{G}} \text{ and } p \in \mathbf{\Lambda}\}. \quad (20)$$

Then, following BFS order of nodes, we retrieve the optimal parse tree at each $p \in \mathbb{P}$: starting from the root Or-node, we select the optimal branch (with the largest score) of each encountered Or-node, keep the two child nodes of each encountered And-node, and retrieve the optimal position of each encountered part terminal-node (by taking $\arg\max$ for the second case in Eqn.(19)).

After spatial parsing, we apply non-maximum suppression (NMS) in computing the optimal parse trees with a predefined intersection-over-union (IoU) overlap threshold, denoted by τ_{NMS}. We keep top N_{best} parse trees to infer the best B_t^* together with a temporal DP algorithm, similar to the strategies used in [19], [20].

Temporal DP: Assuming that all the N-best candidates for $B_2, \cdots, B_t$ are memoized after running spatial DP algorithm in I_2 to I_t, Eqn.(15) corresponds to the classic DP formulation of forward and backward inference for decoding HMMs with $-\text{Score}(I_i|B_i; \mathcal{G})$ being the singleton "data" term and $\text{Cost}(B_i|B_{i-1})$ the pairwise cost term.

Let $\mathbb{B}_i[B_i]$ be energy of the best object states in the first i frames with the constraint that the i-th one is B_i. We have,

$$\begin{aligned} \mathbb{B}_1[B_1] &= -\text{Score}(I_1|B_1; \mathcal{G}), \\ \mathbb{B}_i[B_i] &= -\text{Score}(I_i|B_i; \mathcal{G}) \\ &\quad + \min_{B_{i-1}} (\mathbb{B}_{i-1}[B_{i-1}] + \text{Cost}(B_i|B_{i-1})). \end{aligned} \quad (21)$$

When B_1 is the input bounding box. Then, the temporal DP algorithm consists of two steps:

i) *The forward step* for computing all $\mathbb{B}_i[B_i]$'s, and caching the optimal solution for B_{i-1} as a function of B_i for later back-tracing starting at $i = 2$,

$$\mathbf{T}_i[B_i] = \arg\min_{B_{i-1}} \{\mathbb{B}_{i-1}[B_{i-1}] + \text{Cost}(B_i|B_{i-1})\}.$$

ii) *The backward step* for finding the optimal trajectory $(B_1, B_2^*, \cdots, B_t^*)$, where we first take,

$$B_t^* = \arg\min_{B_t} \mathbb{B}_t[B_t], \quad (22)$$

and then in the order of $i = t-1, \cdots, 2$ trace back,

$$B_i^* = \mathbf{T}_{i+1}[B_{i+1}^*]. \quad (23)$$

In practice, we often do not need to run temporal DP in the whole time range $[1, t]$, especially for long-term tracking, since the target object might have changed significantly or we might have camera motion, instead we only focus on some short time range, $[t - \Delta t, t]$ (see settings in experiments).

Remarks: In our TLP method, we apply the spatial and the temporal DP algorithms in a stage-wise manner and without tracking parts explicitly. Thus, we do not introduce loops in inference. If we instead attempt to learn a joint spatial-temporal AOG, it will be a much more difficult problem due to loops in joint spatial-temporal inference, and approximate inference is used.

Search Strategy: During tracking, at time t, B_t is initialized by B_{t-1}, and then a rectangular region of interest (ROI) centered at the center of B_t is used to compute feature pyramid and run parsing with AOG. The ROI is first computed as a square area with the side length being s_{ROI} times longer than the maximum of width and height of B_t and then is clipped with the image domain. If no candidates are found (i.e., Ω_{cand} is empty), we will run the parsing in whole image domain. So, our AOGTracker is capable of re-detecting a tracked object. If there are still no candidates (e.g., the target object was completely occluded or went out of camera view), the tracking result of this frame is set to be invalid and we do not need to run the temporal DP.

4.3 The Trackability of an Object AOG

To detect critical moments online, we need to measure the quality of an object AOG, $\mathcal{G}$ at time t. We compute its trackability based on the score maps in which the optimal parse tree is placed. For each node v in the parse tree, we have its position in score map pyramid (i.e., the level of pyramid and the location in that level), (l_v, x_v, y_v). We define the trackability of node v by,

$$\text{Trackability}(v|I_t, \mathcal{G}) = S(l_v, x_v, y_v) - \mu_S \quad (24)$$

where $S(l_v, x_v, y_v)$ is the score of node v, μ_S the mean score computed from the whole score map. Intuitively, we expect the score map of a discriminative node v has peak and steep landscape, as investigated in [51]. The trackabilities of part nodes are used to infer partial occlusion and local structure variations, and trackability of the inferred parse tree indicate the "goodness"

of current object AOG. We note that we treat trackability and intrackability (i.e., the inverse of th trackability) exchangeably. More sophisticated definitions of intrackability in tracking are referred to [52].

We model trackability by a Gaussian model whose mean and standard derivation are computed incrementally in $[2, t]$. At time t, a tracked object is said to be "intrackable" if its trackability is less than $mean_{trackability}(t) - 3 \cdot std_{trackability}(t)$. We note that the tracking result could be still valid even if it is "intrackable" (e.g., in the first few frames in which the target object is occluded partially, especially by similar distractors).

5 Online Learning of Object AOGs

In this section, we present online learning of object AOGs, which consists of three components: (i) Maintaining a training dataset based on tracking results; (ii) Estimating parameters of a given object AOG; and (iii) Learning structure of the object AOG by pruning full structure AOG, which requires (ii) in the process.

5.1 Maintaining the Training Dataset Online

Denote by $D_t = D_t^+ \cup D_t^-$ the training dataset at time t, consisting of D_t^+, a positive dataset, and D_t^-, a negative dataset.

In the first frame, we have $D_1^+ = \{(I_1, B_1)\}$ and let $B_1 = (x_1, y_1, w_1, h_1)$. We augment it with eight locally shifted positives, i.e., $\{I_1, B_{1,i}; i = 1, \cdots, 8\}$ where $x_{1,i} \in \{x_1 \pm d\}$ and $y_{1,i} \in \{y_{\pm} d\}$ with width and height not changed. d is set to the cell size in computing HOG features. The initial D_1^- uses the whole remaining image $I_{\overline{\Lambda_{B_1}}}$ for mining hard negatives in training.

At time t, if B_t is valid according to tracking-by-parsing, we have $D_t^+ = D_{t-1}^+ \cup \{(I_t, B_t)\}$, and add to D_t^- all other candidates in Ω_{cand} (Eqn. 20) which are not suppressed by B_t according to NMS (i.e., hard negatives). Otherwise, we have $D_t = D_{t-1}$.

5.2 Estimating Parameters of a Given Object AOG

We use latent SVM method (LSVM) [1]. Based on the scoring functions defined in Section 4.1, we can re-write the scoring function of applying a given object AOG, $\mathcal{G}$ on a training example (denoted by I_B for simplicity),

$$\text{Score}(I_B; \mathcal{G}) = \max_{pt \in \Omega_{\mathcal{G}}} < \Theta, \Phi(\mathbb{F}, pt) > \quad (25)$$

where pt represents a parse tree, $\Omega_{\mathcal{G}}$ the space of parse trees, Θ the concatenated vector of all parameters, $\Phi(\mathbb{F}, pg)$ the concatenated vector of appearance and deformation features in feature pyramid $\mathbb{F}$ w.r.t. parse tree pt, and the bias term.

The objective function in estimating parameters is defined by the l_2-regularized empirical hinge loss function,

$$\mathcal{L}_{D_t}(\Theta) = \frac{1}{2}||\Theta||_2^2 + \frac{C}{|D_t|}[\sum_{I_B \in D_t^+} \max(0, 1 - \text{Score}(I_B; \mathcal{G})) + \sum_{I_B \in D_t^-} \max(0, 1 + \text{Score}(I_B; \mathcal{G}))] \quad (26)$$

where C is the trade-off parameter in learning. Eqn.(26) is a semi-convexity function of the parameters Θ due to the empirical loss term on positives.

In optimization, we utilize an iterative procedure in a "co-ordinate descent" way. We first convert the object function to a convex function by assigning latent values for all positives using the spatial DP algorithm. Then, we estimate parameters. While we can use stochastic gradient descent as done in DPMs [1], we adopt LBFGS method in practice [3] [53] since it is more robust and efficient with parallel implementation as investigated in [9], [54]. The detection threshold, $\tau_{\mathcal{G}}$ is estimated as the minimum score of positives.

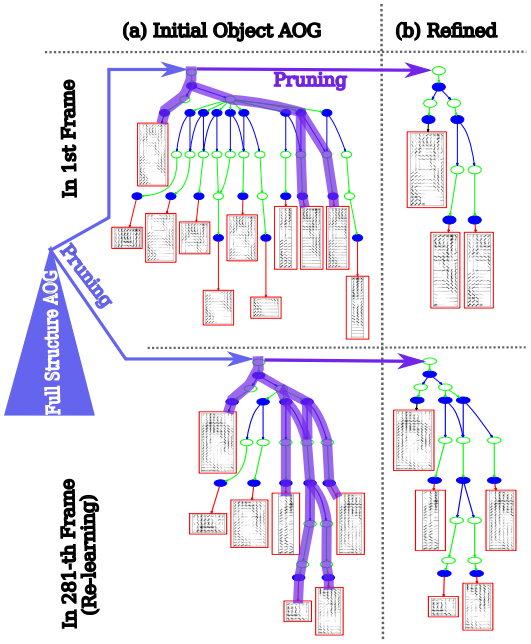


Fig. 7: Illustration of learning an object AOG in the first frame (top) and re-learning an object AOG in the 281-th frame when a critical moment has triggered. It consists of two steps: (a) learning initial object AOG by pruning branches of Or-nodes in full structure AOG, and (b) learning refined object AOG by pruning part configurations w.r.t. majority voting in positive re-labeling in LSVM.

5.3 Learning Object AOGs

With the training dataset D_t and the full structure AOG constructed based on B_1, an object AOG is learned in three steps:

i) Evaluating the figure of merits of nodes in the full structure AOG. We first train the root classifier (i.e., object appearance parameters and bias term) by linear SVM using D_t^+ and data-mining hard negatives in D_t^-. Then, the appearance parameters for each part terminal-node $\mathbf{t}$ is initialized by cropping out the corresponding portion in the object template [4]. Following DFS order, we evaluate the figure of merit of each node in the full structure AOG by its training error rate. The error rate is calculated on D_t where the score of a node is computed w.r.t. scoring functions defined in Section 4.1. The smaller the error rate is, the more discriminative a node is.

ii) Retrieving an initial object AOG and re-estimating parameters. We retrieve the most discriminative subgraph in the full structure AOG as initial object AOG. Following BFS order, we

3. We reimplemented the matlab code available at http://www.cs.ubc.ca/ schmidtm/Software/minConf.html in c++.

4. We also tried to train the linear SVM classifiers for all the terminal-nodes individually using cropped examples, which increases the runtime, but does not improve the tracking performance in experiments. So, we use the simplified method above.

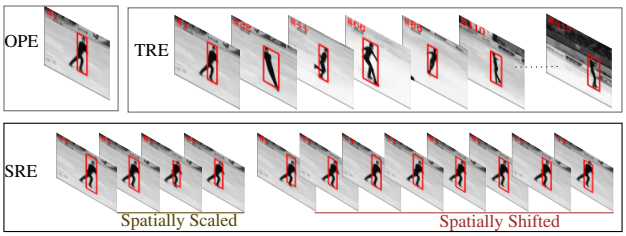


Fig. 8: Illustration of the three types of evaluation methods in TB-100/50/CVPR2013. *In one-pass evaluation (OPE)*, a tracker is initialized in the first frame and let it track the target until the end of the sequence. *In temporal robustness evaluation (TRE)*, a tracker starts at different starting frames initialized with the corresponding ground-truth bounding boxes and then tracks the object until the end. 20 starting frames (including the first frame) are used in TB-100. *In spatial robustness evaluation (SRE)*, a tracker runs multiple times with spatially scaled (4 types) and shifted (8 types of perturbation) initializations in the first frame.

start from the root Or-node, select for each encountered Or-node the best child node (with the smallest training error rate among all children) and the child nodes whose training error rates are not bigger than that of the best child by some predefined small positive value (i.e., preserving ambiguities), keep the two child nodes for each encountered And-node, and stop at each encountered terminal-node. We show two examples in the left of Fig. 7. We train the parameters of initial object AOG using LSVM [1] with two rounds of positive re-labeling and hard negative mining respectively.

iii) Controlling model complexity. To do that, a refined object AOG for tracking is obtained by further selecting the most discriminative part configuration(s) in the initial object AOG learned in the step ii). The selection process is based on latent assignment in relabeling positives in LSVM training. A part configuration in the initial object AOG is pruned if it relabeled less than 10% positives (see the right of Fig. 7). We further train the refined object AOG with one round latent positive re-labeling and hard negative mining. By reducing model complexity, we can speed up the tracking-by-parsing procedure.

Verification of a refined object AOG. We run parsing with a refined object AOG in the first frame. The refined object AOG is accepted if the score of the optimal parse tree is greater than the threshold estimated in training and the IoU overlap between the predicted bounding box and the input bounding box is greater than or equals the IoU NMS threshold, τ_{NMS} in detection.

Identifying critical moments in tracking. A critical moment means a tracker has become "uncertain" and at the same time accumulated "enough" new samples, which is triggered in tracking when two conditions were satisfied. The first is that the number of frames in which a tracked object is "intrackable" was larger than some value, $N_{\text{Intrackable}}$. The second is that the number of new valid tracking results are greater than some value, $N_{\text{NewSample}}$. Both are accumulated from the last time an object AOG was re-learned.

The spatial resolution of placing parts. In learning object AOGs, we first place parts at the same spatial resolution as the object. If the learned object AOG was not accepted in verification, we then place parts at twice the spatial resolution w.r.t. the object and re-learn the object AOG. In our experiments, the two specifications handled all testing sequences successfully.

Overall flow of online learning. In the first frame or when a critical moment is identified in tracking, we learn both structure and parameters of an object AOG, otherwise we update parameters only if the tracking result is valid in a frame based on tracking-by-parsing.

	Representation										Search			
	Local	Template	Color	Histogram	Subspace	Sparse	Binary or Haar	Discriminative	Generative	Model Update	Particle Filter	MCMC	Local Optimum	Dense Sampling
ASLA [55]	✓	✓			✓	✓			✓	✓	✓			
BSBT [56]							H	✓		✓				✓
CPF [57]	✓		✓	✓					✓		✓			
CSK [58]		✓						✓		✓				✓
CT [59]							H	✓		✓				✓
CXT [60]							B	✓		✓				✓
DFT [61]	✓	✓							✓	✓			✓	
FOT [62]	✓	✓							✓	✓			✓	
FRAG [63]	✓			✓					✓					✓
IVT [29]		✓			✓				✓	✓	✓			
KMS [30]			✓	✓					✓				✓	✓
L1APG [64]		✓			✓	✓			✓	✓	✓			
LOT [65]	✓		✓						✓	✓	✓			
LSHT [66]	✓		✓	✓			H		✓	✓				✓
LSK [67]	✓	✓			✓	✓			✓				✓	
LSS [68]	✓	✓			✓	✓			✓	✓	✓			
MIL [39]							H	✓		✓				✓
MTT [69]	✓			✓	✓			✓	✓	✓				
OAB [70]							H	✓		✓				✓
ORIA [71]		✓			✓		H	✓		✓				✓
PCOM [72]	✓			✓	✓			✓	✓	✓				
SCM [73]	✓	✓			✓	✓		✓	✓	✓	✓			✓
SMS [74]			✓	✓					✓				✓	
SBT [75]							H	✓		✓				✓
STRUCK [40]							H	✓		✓				✓
TLD [17]	✓						B	✓		✓				✓
VR [76]			✓					✓		✓			✓	
VTD [77]		✓	✓		✓				✓	✓		✓		
VTS [78]	✓	✓	✓		✓				✓	✓		✓		
AOG	✓	✓	✓	✓			HOG [+Color]	✓		✓			✓	✓

TABLE 2: Tracking algorithms evaluated in the TB-100 benchmark (reproduced from [2]).

6 Experiments

In this section, we present comparison results on the TB-50/100/CVPR2013 benchmarks [2], [3] and the VOT benchmarks [4]. We also analyze different aspects of our method. The source code [5] is released with this paper for reproducing all results. We denote the proposed method by *AOG* in tables and plots.

Parameter Setting. We use the same parameters for all experiments since we emphasize online learning in this paper. In learning object AOGs, the side length of the grid used for constructing the full structure AOG is either 3 or 4 depending the slide length of input bounding box (to reduce the time complexity of online learning). The number of intervals in computing feature pyramid is set to 6 with cell size being 4. The factor s in computing search ROI is set to $s_{\text{ROI}} = 3$. The NMS IoU threshold is set to $\tau_{\text{NMS}} = 0.7$. The number of top parse trees kept after spatial DP parsing is set $N_{\text{Best}} = 10$. The time range in temporal DP algorithm is set to $\Delta t = 5$. In identifying critical moments, we set $N_{\text{Intrackable}} = 5$ and $N_{\text{NewSample}} = 10$. The LSVM trade-off parameter in Eqn.(26) is set to $C = 0.001$. When re-learning structure and parameters, we could use all the frames with valid tracking results. To reduce the time complexity, the number of frames used in re-learning is at most 100 in our experiments. At time t, we first take the first 10 frames with valid tracking results in $[1, t]$ with the underlying intuition that they have high probabilities of being tracked correctly (note that we alway use the first frame since the ground-truth bounding box is given), and then take the remaining frames in reversed time order.

Speed. In our current c++ implementation, we adopt FFT in computing score pyramids as done in [54] which also utilizes

5. Available at https://github.com/tfwu/RGM-AOGTracker

Metric	Success Rate / Precision Rate								
Evaluation	OPE			SRE			TRE		
Subset	100	50	CVPR2013	100	50	CVPR2013	100	50	CVPR2013
AOG Gain	13.93 / 18.06	16.84 / 22.23	2.74 / 19.37	11.47 / 16.79	12.52 / 17.82	11.89 / 17.55	9.25 / 11.06	11.37 / 14.61	11.59 / 14.38
Runner-up	STRUCK [40]		SO-DLT [6] / STRUCK [40]	STRUCK [40]					

Subsets in TB-50	DEF(23)	FM(25)	MB(19)	IPR(29)	BC(20)	OPR(32)	OCC(29)	IV(22)	LR(8)	SV(38)	OV(11)
AOG Gain (success rate)	15.89	15.56	17.29	12.29	17.81	14.04	14.7	15.73	6.65	18.38	15.99
Runner-up	STRUCK [40]		TLD [17]		SCM [73]						MIL [39]

TABLE 3: Performance gain (in %) of our AOGTracker in term of success rate and precision rate in the benchmark [2]. Success plots of TB-100/50/CVPR2013 are shown in Fig. 9. Precision plots of TB-100/50/CVPR2013, and success and precision plots in the 11 subsets of TB-100/50/CVPR2013 are provided in the supplementary material due to space limit here.

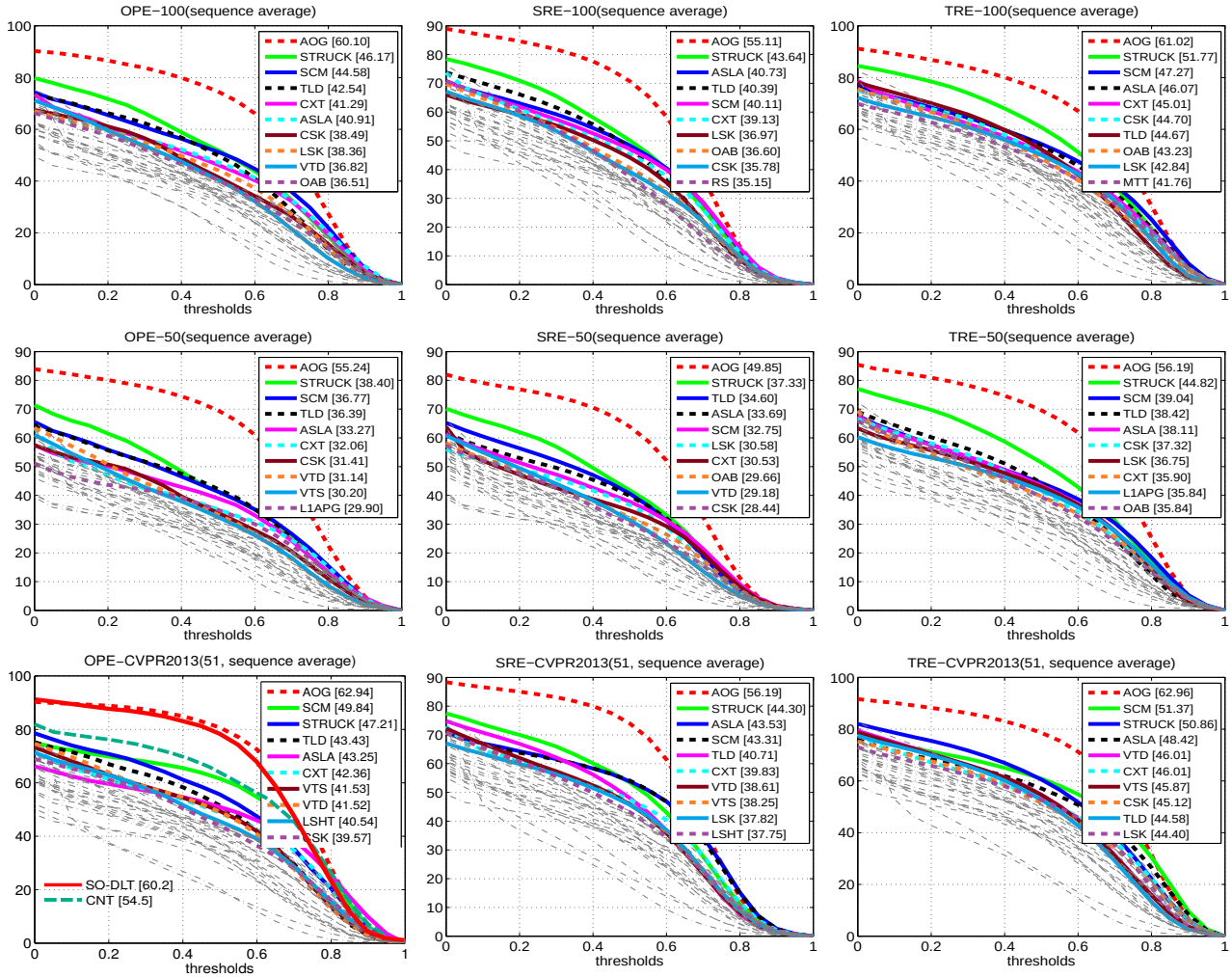


Fig. 9: Performance comparison in TB-100 (1st row), TB-50 (2nd row) and TB-CVPR2013 (3rd row) in term of success plots of OPE (1st column), SRE (2nd column) and TRE (3rd colum). For clarity, only top 10 trackers are shown in color curves and listed in the legend. Two deep learning based trackers, CNT [5] and SO-DLT [6], are evaluated in TB-CVPR2013 using OPE (with their performance plots manually added in the left-bottom figure). *We note that the plots are reproduced with the raw results provided at http://cvlab.hanyang.ac.kr/tracker_benchmark/.* (Best viewed in color and with magnification)

multi-threads with OpenMP. We also provide a distributed version based on MPI [6] in evaluation. The FPS is about 2 to 3. We are experimenting GPU implementations to speed up our TLP.

6.1 Results on TB-50/100/CVPR2013

The TB-100 benchmark has 100 target objects ($58,897$ frames in total) with 29 publicly available trackers evaluated. It is extended from a previous benchmark with 51 target objects released at CVPR2013 (denoted by TB-CVPR2013). Further, since some target objects are similar or less challenging, a subset of 50 difficult and representative ones (denoted by TB-50) is selected for an in-depth analysis. Two types of performance metric are used, the **precision plot** (i.e., the percentage of frames in which estimated locations are within a given threshold distance of ground-truth positions) and the **success plot** (i.e., based on IoU overlap scores which are commonly used in object detection benchmarks, e.g., PASCAL VOC [79]). The higher a success rate or a precision rate

6. https://www.mpich.org/

Fig. 10: Qualitative results. For clarity, we show tracking results (bounding boxes) in 6 randomly sampled frames for the top 10 trackers according to their OPE performance in TB-100. (Best viewed in color and with magnification.)

is, the better a tracker is. Usually, success plots are preferred to rank trackers [2], [4] (thus we focus on success plots in comparison). Three types of evaluation methods are used as illustrated in Fig.8.

To account for different factors of a test sequence affecting performance, the testing sequences are further categorized w.r.t. 11 attributes for more ind-depth comparisons: (1) Illumination Variation (IV, 38/22/21 sequences in TB-100/50/CVPR2013), (2) Scale Variation (SV, 64/38/28 sequences), (3) Occlusion (OCC, 49/29/29 sequences), (4) Deformation (DEF, 44/23/19 sequences), (5) Motion Blur (MB, 29/19/12 sequences), (6) Fast Motion (FM, 39/25/17 sequences), (7) In-Plane Rotation (IPR, 51/29/31 sequences), (8) Out-of-Plane Rotation (OPR, 63/32/39 sequences), (9) Out-of-View (OV, 14/11/6 sequences), (10) Background Clutters (BC, 31/20/21 sequences), and (11) Low Resolution (LR, 9/8/4 sequences). More details on the attributes and their distributions in the benchmark are referred to [2], [3].

Table. 2 lists the 29 evaluated tracking algorithms which are categorized based on representation and search scheme. See more details about categorizing these trackers in [2]. In TB-CVPR2013, two recent trackers trained by deep convolutional network (CNT [5], SO-DLT [6]) were evaluated using OPE.

We summarize the performance gain of our AOGTracker in Table.3. Our AOGTracker obtains significant improvement (more than 12%) in the 10 subsets in TB-50. Our AOGTracker handles out-of-view situations much better than other trackers since it is capable of re-detecting target objects in the whole image, and it performs very well in the scale variation subset (see examples in the second and fourth rows in Fig. 10) since it searches over feature pyramid explicitly (with the expense of more computation). Our AOGTracker obtains the least improvement in the low-resolution subset since it uses HOG features and the discrepancy between HOG cell-based coordinate and pixel-based one can cause some loss in overlap measurement, especially in the low resolution subset. We will add automatic selection of feature types (e.g., HOG v.s. pixel-based features such as intensity and gradient) according to the resolution, as well as other factors in future work.

Fig.9 shows success plots of OPE, SRE and TRE in TB-100/50/CVPR2013. Our AOGTracker consistently outperforms all other trackers. We note that for OPE in TB-CVPR2013, although the improvement of our AOGTracker over the SO-DLT [6] is not very big, the SO-DLT utilized two deep convolutional networks

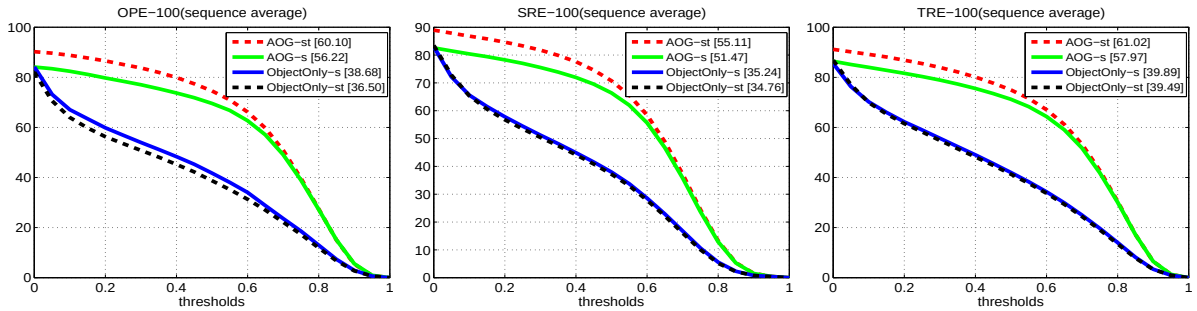


Fig. 11: Performance comparison of four variants of our AOGTracker in TB-100 using success plots of OPE, SRE and TRE.

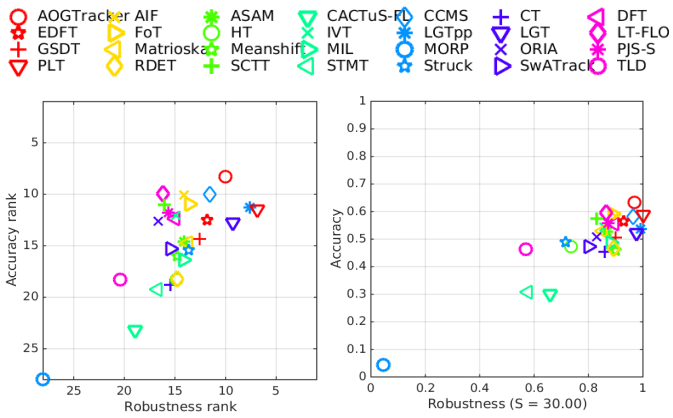


Fig. 12: Performance comparison in VOT2013. *Left*: Ranking plot for the baseline experiment. The smaller the rank number is, the better a tracker is w.r.t. accuracy and/or robust (i.e., the right-top region indicates better performance) *Right*: Accuracy-Robustness plot. The larger the rate is, the better a tracker is.

with different model update strategies in tracking, both of which are pretrained on the ImageNet [34]. Fig. 10 shows some qualitative results.

6.2 Analyses of AOG models and the TLP Algorithm

To show advantage of the proposed method, we compare performance of four different variants of our AOGTracker.

AOG-s vs AOG-st. As stated above, in our AOGTracker, we use a very simple setting for temporal DP which takes into account $\Delta t = 5$ frames, $[t-5, t]$ in our experiments. We denote it by *AOG-st* in comparison. We show the advantage of this simple spatial and temporal DP algorithms by comparing it with the baseline tracker, denoted by *AOG-s*, which uses spatial DP only. Everything else in online inference and learning is the same as AOG-st.

Single object template vs AOG. We test our tracker without learning object part configurations, that is to learn and update object templates only in tracking. Similarly, we also have two settings, with temporal DP (denoted by *ObjectOnly-st*) and without temporal DP (denoted by *ObjectOnly-s*).

Fig. 11 shows performance comparison of the four variants. The full method (AOG-st) obtains the best performance consistently and the two trackers with AOGs significantly outperforms the other two variants. For the two trackers with object templates only, the one without temporal DP (*ObjectOnly-s*) slightly outperform the one with temporal DP (*ObjectOnly-st*), which shows that we might need strong enough object models in integrating spatial and temporal information for better performance.

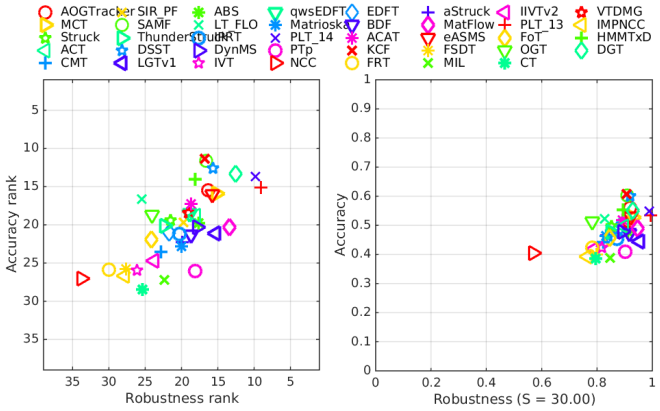


Fig. 13: Performance comparison in VOT2014.

6.3 Results on VOT

In VOT, the evaluation focuses on short-term tracking (i.e., a tracker is not expected to perform re-detection after losing a target object), so the evaluation toolkit will re-initialize a tracker after it loses the target (w.r.t. the condition the overlap between the predicted bounding box and the ground-truth one drops to zero) with the number of failures counted. In VOT protocol, a tracker is tested on each sequence multiple times. The performance is measured in terms of accuracy and robustness. **Accuracy** is computed as the average of per-frame accuracies which themselves are computed by taking the average over the repetitions. **Robustness** is computed as the average number of failure times over repetitions.

We integrate our AOGTracker in the latest VOT toolkit[7] to run experiments with the baseline protocol and to generate plots [8].

The VOT2013 dataset [80] has 16 sequences which was selected from a large pool such that various visual phenomena like occlusion and illumination changes, were still represented well within the selection. 7 sequences are also used in TB-100. There are 27 trackers evaluated. The readers are referred to the VOT technical report [80] for details.

Fig.12 shows the ranking plot and AR plot in VOT2013. Our AOGTracker obtains the best accuracy while its robustness is slightly worse than three other trackers (i.e., PLT [80], LGT [82] and LGTpp [83], and PLT was the winner in VOT2013 challenge). Our AOGTracker obtains the best overall rank.

The VOT2014 dataset [81] has 25 sequences extended from VOT2013. The annotation is based on rotated bounding box

7. Available at https://github.com/votchallenge/vot-toolkit, version 3.2

8. The plots for VOT2013 and 2014 might be different compared to those in the original VOT reports [80], [81] due to the new version of vot-toolkit.

instead of up-right rectangle. There are 33 trackers evaluated. Details on the trackers are referred to [81]. Fig.13 shows the ranking plot and AR plot. Our AOGTracker is comparable to other trackers. One main limitation of AOGTracker is that it does not handle rotated bounding boxes well.

The VOT2015 dataset [84] consists of 60 short sequences (with rotated bounding box annotations) and VOT-TIR2015 comprises 20 sequences (with bounding box annotations). There are 62 and 28 trackers evaluated in VOT2015 and VOT-TIR2015 respectively. Our AOGTracker obtains 51% and 65% (tied for third place) in accuracy in VOT2015 and VOT-TIR2015 respectively. The details are referred to the reports [84] due to space limit here.

7 Discussion and Future Work

We have presented a tracking, learning and parsing (TLP) framework and derived a spatial dynamic programming (DP) and a temporal DP algorithm for online object tracking with AOGs. We also have presented a method of online learning object AOGs including its structure and parameters. In experiments, we test our method on two main public benchmark datasets and experimental results show better or comparable performance.

In our ongoing and future work, we are extending the TLP framework by incorporating generic category-level AOGs [8] to scale up the TLP framework. The generic AOGs are pre-trained offline (e.g., using the PASCAL VOC [79] or the imagenet [34]), and will help the online learning of specific AOGs for a target object (e.g., help to maintain the purity of the positive and negative datasets collected online). The generic AOGs will also be updated online together with the specific AOGs. By integrating generic and specific AOGs, we aim at the life-long learning of objects in videos without annotations. Furthermore, we are also interested in integrating scene grammar [85] and event grammar [86] to leverage more top-down information.

Acknowledgments

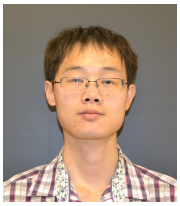
This work is supported by the DARPA SIMPLEX Award N66001-15-C-4035, the ONR MURI grant N00014-16-1-2007, and NSF IIS-1423305. We thank Steven Holtzen for proofreading this paper. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of one GPU.

References

[1] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.
[2] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *PAMI*, vol. 37, no. 9, pp. 1834–1848, 2015.
[3] ——, "Online object tracking: A benchmark," in *CVPR*, 2013.
[4] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. P. Pflugfelder, G. Fernández, G. Nebehay, F. Porikli, and L. Cehovin, "A novel performance evaluation methodology for single-target trackers," *CoRR*, vol. abs/1503.01313, 2015. [Online]. Available: http://arxiv.org/abs/1503.01313
[5] K. Zhang, Q. Liu, Y. Wu, and M.-H. Yang, "Robust visual tracking via convolutional networks," *arXiv preprint arXiv:1501.04505v2*, 2015.
[6] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung, "Transferring rich feature hierarchies for robust visual tracking," *arXiv preprint arXiv:1501.04587v2*, 2015.
[7] S. Carey, *The Origin of Concepts*. Oxford University Press, 2011.
[8] X. Song, T. Wu, Y. Jia, and S.-C. Zhu, "Discriminatively trained and-or tree models for object detection," in *CVPR*, 2013.
[9] R. Girshick, P. Felzenszwalb, and D. McAllester, "Object detection with grammar models," in *NIPS*, 2011.
[10] P. Felzenszwalb and D. McAllester, "Object detection grammars," University of Chicago, Computer Science TR-2010-02, Tech. Rep., 2010.
[11] S. C. Zhu and D. Mumford, "A stochastic grammar of images," *Foundations and Trends in Computer Graphics and Vision*, vol. 2, no. 4, pp. 259–362, 2006.
[12] Y. Amit and A. Trouvé, "POP: patchwork of parts models for object recognition," *IJCV*, vol. 75, no. 2, pp. 267–282, 2007. [Online]. Available: http://dx.doi.org/10.1007/s11263-006-0033-9
[13] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, 2006.
[14] L. R. Rabiner, "Readings in speech recognition," A. Waibel and K.-F. Lee, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pp. 267–296. [Online]. Available: http://dl.acm.org/citation.cfm?id=108235.108253
[15] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *IJCV*, vol. 29, no. 1, pp. 5–28, 1998.
[16] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *CVPR*, 2008.
[17] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *PAMI*, vol. 34, no. 7, pp. 1409–1422, 2012.
[18] J. S. S. III and D. Ramanan, "Self-paced learning for long-term tracking," in *CVPR*, 2013.
[19] D. Park and D. Ramanan, "N-best maximal decoder for part models," in *ICCV*, 2011.
[20] D. Batra, P. Yadollahpour, A. Guzmán-Rivera, and G. Shakhnarovich, "Diverse m-best solutions in markov random fields," in *ECCV*, 2012.
[21] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *CVPR*, 2008.
[22] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *CVPR*, 2011.
[23] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *PAMI*, vol. 33, no. 9, pp. 1806–1819, 2011.
[24] A. V. Goldberg, "An efficient implementation of a scaling minimum-cost flow algorithm," *J. Algorithms*, vol. 22, no. 1, pp. 1–29, 1997.
[25] S. Hong and B. Han, "Visual tracking by sampling tree-structured graphical models," in *ECCV*, 2014.
[26] S. Hong, S. Kwak, and B. Han, "Orderless tracking through model-averaged posterior estimation," in *ICCV*, 2013.
[27] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. van den Hengel, "Part-based visual tracking with online latent structural learning," in *CVPR*, 2013.
[28] H. Nam, S. Hong, and B. Han, "Online graph-based tracking," in *ECCV*, 2014.
[29] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *IJCV*, vol. 77, no. 1-3, pp. 125–141, 2008.
[30] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *PAMI*, vol. 25, no. 5, pp. 564–575, 2003.
[31] X. Mei and H. Ling, "Robust visual tracking and vehicle classification via sparse representation," *PAMI*, vol. 33, no. 11, pp. 2259–2272, 2011.
[32] X. Li, A. R. Dick, C. Shen, A. van den Hengel, and H. Wang, "Incremental learning of 3d-dct compact representations for robust visual tracking," *PAMI*, vol. 35, no. 4, pp. 863–881, 2013.
[33] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *CVPR*, 2016.
[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.
[35] J. Kwon and K. M. Lee, "Highly nonrigid object tracking via patch-based dynamic appearance modeling," *PAMI*, vol. 35, no. 10, pp. 2427–2441, 2013.
[36] L. Cehovin, M. Kristan, and A. Leonardis, "Robust visual tracking using an adaptive coupled-layer visual model," *PAMI*, vol. 35, no. 4, pp. 941–953, 2013.
[37] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *CVPR*, 2012.
[38] S. Avidan, "Support vector tracking," *PAMI*, vol. 26, no. 8, pp. 1064–1072, 2004.
[39] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *PAMI*, vol. 33, no. 8, pp. 1619–1632, 2011.
[40] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *ICCV*, 2011.
[41] J. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *ECCV*, 2012.
[42] V. Mahadevan and N. Vasconcelos, "Biologically inspired object tracking using center-surround saliency mechanisms," *PAMI*, vol. 35, no. 3, pp. 541–554, 2013.

[43] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. van den Hengel, "Part-based visual tracking with online latent structural learning," in *CVPR*, 2013.
[44] L. Zhang and L. van der Maaten, "Structure preserving object tracking," in *CVPR*, 2013.
[45] J. Shi and C. Tomasi, "Good feature to track," in *CVPR*, 1994.
[46] Y. Lu, T. Wu, and S.-C. Zhu, "Online object tracking, learning and parsing with and-or graphs," in *CVPR*, 2014.
[47] X. Li, W. Hu, C. Shen, Z. Zhang, A. R. Dick, and A. van den Hengel, "A survey of appearance models in visual object tracking," *CoRR*, vol. abs/1303.4803, 2013. [Online]. Available: http://arxiv.org/abs/1303.4803
[48] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *IJCV*, vol. 56, no. 3, pp. 221–255, 2004.
[49] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.
[50] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in *ICPR*, 1994.
[51] J. Kwon and K. M. Lee, "Highly nonrigid object tracking via patch-based dynamic appearance modeling," *TPAMI*, vol. 35, no. 10, pp. 2427–2441, 2013.
[52] H. Gong and S. C. Zhu, "Intrackability: Characterizing video statistics and pursuing video representations," *IJCV*, vol. 97, no. 3, pp. 255–275, 2012.
[53] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Comput.*, vol. 16, no. 5, pp. 1190–1208, 1995.
[54] C. Dubout and F. Fleuret, "Exact acceleration of linear object detectors," in *ECCV*, 2012.
[55] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *CVPR*, 2012.
[56] S. Stalder, H. Grabner, and L. van Gool, "Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition," in *ICCV Workshop*, 2009.
[57] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *ECCV*, 2002.
[58] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *ECCV*, 2012.
[59] K. Zhang, L. Zhang, and M. Yang, "Fast compressive tracking," *PAMI*, vol. 36, no. 10, pp. 2002–2015, 2014.
[60] T. B. Dinh, N. Vo, and G. G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments," in *CVPR*, 2011.
[61] L. Sevilla-Lara and E. Learned-Miller, "Distribution fields for tracking," in *CVPR*, 2012.
[62] T. Vojir and J. Matas, "Robustifying the flock of trackers," in *Computer Vision Winter Workshop*, 2011.
[63] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *CVPR*, 2006.
[64] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust L1 tracker using accelerated proximal gradient approach," in *CVPR*, 2012.
[65] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," in *CVPR*, 2012.
[66] S. He, Q. Yang, R. W. Lau, J. Wang, and M.-H. Yang, "Visual tracking via locality sensitive histograms," in *CVPR*, 2013.
[67] B. Liu, J. Huang, L. Yang, and C. A. Kulikowski, "Robust tracking using local sparse appearance model and k-selection," in *CVPR*, 2011.
[68] D. Wang, H. Lu, and M.-H. Yang, "Least soft-thresold squares tracking," in *CVPR*, 2013.
[69] T.Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *CVPR*, 2012.
[70] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *BMVC*, 2006.
[71] Y. Wu, B. Shen, and H. Ling, "Online robust image alignment via iterative convex optimization," in *CVPR*, 2012.
[72] D. Wang and H. Lu, "Visual tracking via probability continuous outlier model," in *CVPR*, 2014.
[73] W. Zhong, H. Lu, and M. Yang, "Robust object tracking via sparsity-based collaborative model," in *CVPR*, 2012.
[74] R. T. Collins, "Mean-shift blob tracking through scale space," in *CVPR*, 2003.
[75] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *ECCV*, 2008.
[76] R. T. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *PAMI*, vol. 27, no. 10, pp. 1631–1643, 2005.
[77] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *CVPR*, 2010.
[78] ——, "Tracking by sampling trackers," in *ICCV*, 2011.
[79] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results."
[80] M. Kristan and et al, "The visual object tracking vot2013 challenge results," 2013. [Online]. Available: http://www.votchallenge.net/vot2013/program.html
[81] ——, "The visual object tracking vot2014 challenge results," 2014. [Online]. Available: http://www.votchallenge.net/vot2014/program.html
[82] L. Cehovin, M. Kristan, and A. Leonardis, "Robust visual tracking using an adaptive coupled-layer visual model," *PAMI*, vol. 35, no. 4, pp. 941–953, 2013.
[83] J. Xiao, R. Stolkin, and A. Leonardis, "An enhanced adaptive coupled-layer lgtracker++," in *Vis. Obj. Track. Challenge VOT2013, In conjunction with ICCV2013*, 2013.
[84] M. Kristan and et al, "The visual object tracking vot2015 and tir2015 challenge results," 2015. [Online]. Available: http://www.votchallenge.net/vot2015/program.html
[85] Y. Zhao and S. C. Zhu, "Image parsing with stochastic scene grammar," in *NIPS*, 2011.
[86] M. Pei, Z. Si, B. Z. Yao, and S. Zhu, "Learning and parsing video events with goal and intent prediction," *CVIU*, vol. 117, no. 10, pp. 1369–1383, 2013.

Tianfu Wu received Ph.D. degree in Statistics from University of California, Los Angeles (UCLA) in 2011. He is currently research assistant professor in the center for vision, cognition, learning and autonomy (VCLA) at UCLA. His research interests include: (i) Statistical learning of large scale hierarchical and compositional models (e.g., And-Or graphs) from images and videos. (ii) Statistical inference by learning near-optimal cost-sensitive decision policies. (iii) Statistical theory of performance guaranteed learning algorithm and inference procedure.



Yang Lu is currently Ph. D. student in the Center for Vision, Cognition, Learning and Autonomy at the University of California, Los Angeles. He received B.S. degree and M.S. degree in Computer Science from Beijing Institute of Technology, China, in 2009 and in 2012 respectively. He received the University Fellowship from UCLA and National Fellowships from Department of Education at China. His current research interests include Computer Vision and Statistical Machine Learning. Specifically, his research interests focus on statistical modeling of natural images and videos, and structure learning of hierarchical models.

Song-Chun Zhu received Ph.D. degree from Harvard University in 1996. He is currently professor of Statistics and Computer Science at UCLA, and director of Center for Vision, Cognition, Learning and Autonomy. He received a number of honors, including the J.K. Aggarwal prize from the Int'l Association of Pattern Recognition in 2008 for "contributions to a unified foundation for visual pattern conceptualization, modeling, learning, and inference", the David Marr Prize in 2003 with Z. Tu et al. for image parsing, twice Marr Prize honorary nominations in 1999 for texture modeling and in 2007 for object modeling with Z. Si and Y.N. Wu. He received the Sloan Fellowship in 2001, a US NSF Career Award in 2001, and an US ONR Young Investigator Award in 2001. He received the Helmholtz Test-of-time award in ICCV 2013, and he is a Fellow of IEEE since 2011.